Trembr: Exploring Road Networks for Trajectory Representation Learning

TAO-YANG FU and WANG-CHIEN LEE, The Pennsylvania State University, USA

In this article, we propose a novel representation learning framework, namely *TRajectory EMBedding via Road networks (Trembr)*, to learn *trajectory embeddings* (low-dimensional feature vectors) for use in a variety of trajectory applications. The novelty of Trembr lies in (1) the design of a recurrent neural network–(RNN) based encoder–decoder model, namely *Traj2Vec*, that encodes spatial and temporal properties inherent in trajectories into trajectory embeddings by exploiting the underlying road networks to constrain the learning process in accordance with the matched road segments obtained using road network matching techniques (e.g., Barefoot [24, 27]), and (2) the design of a neural network–based model, namely *Road2Vec*, to learn *road segment embeddings* in road networks that captures various relationships amongst road segments in preparation for trajectory representation learning. In addition to model design, several unique technical issues raising in Trembr, including data preparation in Road2Vec, the road segment relevance-aware loss, and the network topology constraint in Traj2Vec, are examined. To validate our ideas, we learn trajectory embeddings using multiple large-scale real-world trajectory datasets and use them in three tasks, including trajectory similarity measure, travel time prediction, and destination prediction. Empirical results show that Trembr soundly outperforms the state-of-the-art trajectory representation learning models, *trajectory2vec* and *t2vec*, by at least one order of magnitude in terms of mean rank in trajectory similarity measure, 23.3% to 41.7% in terms of mean absolute error (MAE) in travel time prediction, and 39.6% to 52.4% in terms of MAE in destination prediction.

CCS Concepts: • **Computing methodologies** → **Learning latent representations**; *Neural networks*; **Multi-task learning**; Regularization;

Additional Key Words and Phrases: Trajectory, neural networks, representation learning, road network

ACM Reference format:
Tao-Yang Fu and Wang-Chien Lee. 2020. Trembr: Exploring Road Networks for Trajectory Representation Learning. *ACM Trans. Intell. Syst. Technol.* 11, 1, Article 10 (February 2020), 25 pages.
https://doi.org/10.1145/3361741

1 INTRODUCTION

With the rapid growth of GPS-enabled devices and the tremendous demands of location-aware applications, enormous amounts of trajectory data are being generated at an unprecedented speed. A trajectory, typically represented as a sequence of spatio-temporal points to describe the movement of a mobile user over time, can be used for various kinds of prediction tasks, e.g., travel time estimation, destination prediction, and trajectory outlier detection [23, 28, 40]. Moreover, trajectory

This work is supported in part by the National Science Foundation under Grant No. IIS-1717084.
Authors' addresses: T.-Y. Fu, W376 Westgate Building, Dept. of Computer Science and Engineering, The Pennsylvania State University, University Park, PA, 16802, USA; email: txf225@psu.edu; W.-C. Lee, W332 Westgate Building, Dept. of Computer Science and Engineering, The Pennsylvania State University, University Park, PA, 16802, USA; email: wul2@psu.edu.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
2157-6904/2020/02-ART10 $15.00
https://doi.org/10.1145/3361741

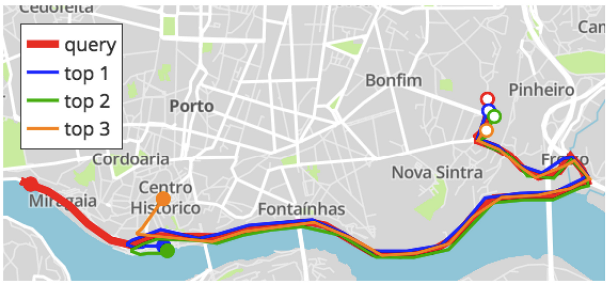


Fig. 1. An example of top three similar trajectories.

data are essential for deriving trajectory similarity measures and exploratory trajectory mining tasks, such as similar trajectory search and trajectory clustering [20, 21]. To achieve good performance in these tasks, a proper representation of trajectories that captures the moving behaviors of those mobile users is much needed, as it may serve as input features to various machine learning or data mining algorithms. Typical approaches for feature engineering usually involve domain experts to manually extract discriminative features that capture inherent characteristics of interested objects for use in specific prediction tasks. These approaches, heavily relying on prior knowledge and experiences of domain experts, are, however, time consuming and cost expensive. Moreover, those specially designed features may not be applicable for other tasks. These issues have inspired emerging research interests in *trajectory representation learning* in recent years [18, 44].

The goal of trajectory representation learning is to automatically capture the moving behaviors of mobile users embedded in the trajectories and encode the information in forms of general-purpose low-dimensional feature vectors, called *trajectory embeddings*, which can be used for various applications.[1] This setting is similar to that of other representation learning research, including word embedding learning, e.g., Word2Vec [25], and network embedding learning, e.g., DeepWalk [31], where the learned embeddings are used for various applications, including document classification, opinion mining in natural language processing, node classification, link prediction in network analytics, and so on. Generally speaking, two spatio-temporally similar trajectories would have embeddings closely located in the latent feature space. Thus, trajectory embeddings serve well to measure trajectory similarity (e.g., by their Euclidean distance in the latent feature space) and as input features for various prediction and mining tasks on trajectory data, such as the aforementioned travel time estimation and destination prediction. As a result, it is beneficial to have general-purpose, precomputed trajectory embeddings for those tasks, as the time and cost spent on the labor-intensive feature engineering effort could be significantly reduced. Figure 1 shows the feasibility and effectiveness of using trajectory embeddings generated in this work to measure trajectory similarity in trajectory search. Given a query trajectory (in red), the top three most similar trajectories (in terms of the Euclidean distance between their trajectory embeddings) are returned. We can see that the three results are very similar with the query.

To learn trajectory embeddings, it is natural to consider *recurrent neural network– (RNN) based encoder–decoders*, e.g., RNN autoencoder and seq2seq [5, 37], which have received great success in modeling sequence data in natural language processing [29] and in image/video processing [35]. In these applications, an RNN-based encoder–decoder encodes an input sequence, e.g., a textual document, into a low-dimensional latent vector that in turn is decoded back to the original sequence to embed the inherent properties of the sequence into a latent vector. However, simply applying

[1]In this work, we use the terms “embeddings,” “latent feature vectors,” and “representations” interchangeably.

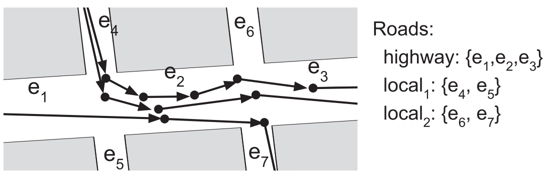


Fig. 2. Trajectories on road network.

the existing RNN-based encoder–decoders on raw trajectory data for trajectory representation learning is impractical. First, raw trajectory data typically suffer from low and non-uniform sampling rates and noisy sample points [45]. Existing RNN-based encoder–decoders, not designed to handle these issues, are not able to capture the real moving path underlying a trajectory (i.e., the exact path taken by a mobile user) into trajectory embeddings. Second, trajectories inherently contain significant spatial and temporal information, but the optimization functions of existing RNN-based encoder–decoders designed for natural language processing or image/video processing do not consider the spatial and temporal properties amongst trajectories. Therefore, those RNN-based encoder–decoders may not truly capture the moving behaviors of mobile users. Third, the movement of a mobile user is physically constrained by factors such as the topology of an underlying road network. However, it is proven that RNN models hardly learn the topology information well automatically [43] and thus fail to optimize the learning of trajectory embeddings.

To address the aforementioned issues, in this article, we propose to encode both the spatial and temporal properties inherent in trajectories into embeddings and incorporate underlying road networks to facilitate and constrain the learning process. A road network is described as a directed graph consisting of intersections as nodes and road segments as edges. With a road network, we assume that the movement of a mobile user, such as a vehicle, is constrained by the topological structure of the road network. Therefore, a trajectory can be seen as describing a moving path on the road network. Different from a typical trajectory represented as a sequence of raw spatio-temporal sample points on roads, we first transform a trajectory as a sequence of road segments coupled with corresponding travel time, called *Spatio-Temporal Sequence (ST-Seq)*, by road network matching techniques (e.g., Barefoot [24, 27]), which map the sample points of a trajectory onto the road network. The ST-Seq, capturing both the underlying moving path and the temporal aspect of the movement, allows the impact of noisy and missing sample points caused by low and non-uniform sampling rates be suppressed. With the ST-Seqs, a new RNN-based encoder–decoder, called *Traj2Vec*, is designed for learning the embeddings of trajectories. Specifically, the proposed Traj2Vec takes an ST-Seq as the input to encode as well as the target to decode. By encoding and decoding both road segments and their corresponding travel times of an ST-Seq jointly, this multi-task learning process of Traj2Vec captures the spatial and temporal behaviors of trajectories. Moreover, Traj2Vec incorporates the topology of the road network into the model to constraint the decoding to guide the learning process.

For the road segments in ST-Seqs, we argue that they are not independent from each other but relevant in some ways due to various relationships, e.g., the same road types and frequent co-occurrence in trajectories. Simply using unique IDs to represent road segments is not able to provide the informative relationships mentioned above. For example, as shown in Figure 2, consider a road network consisting of several road segments belonging to three roads, $highway = \{e_1, e_2, e_3\}$, $local_1 = \{e_4, e_5\}$, and $local_2 = \{e_6, e_7\}$, where three trajectories pass though them. Among these road segments, while e_1, e_2, and e_4 are physically connected and thus relevant, e_2 and e_4 are more relevant to each other than e_1 and e_2 from the aspect of user driving behaviors, because more trajectories go though both e_2 and e_4 than e_1 and e_2. However, from the aspect of

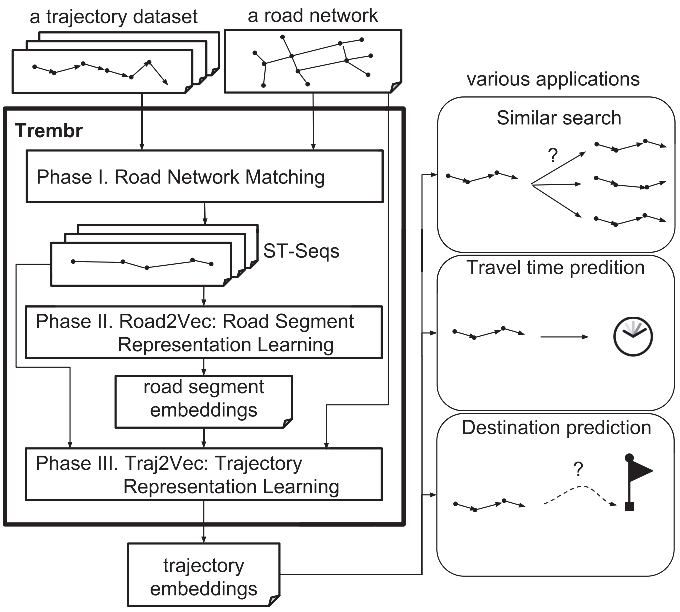


Fig. 3. Overview of the Trembr framework.

road type, e_1 and e_2 are more relevant than e_2 and e_4, because e_1 and e_2 both belong to *highway* and thus have the same road type. To exploit the relevance among road segments, we propose a novel road segment representation learning model, namely *Road segment to Vector (Road2Vec)*, to learn road segment embeddings for use in Traj2Vec, by exploring the above-mentioned relationships between road segments.

Based on the above ideas, we propose a framework, namely *TRajectory EMBedding via Road networks (Trembr)*, for trajectory representation learning. As shown in Figure 3, Trembr performs the following tasks: (1) *trajectory to road network mapping*: mapping each raw trajectory onto the road network as a sequence of (road segment, travel time) pairs, called *ST-Seq*, to address the issues of low and non-uniform sampling, and noisy sample points in raw trajectories; (2) *road segment representation learning*: Road2Vec captures two relationships amongst road segments, i.e., frequent co-occurrence in trajectories and having the same road type, to generate road segment embeddings; and (3) *trajectory representation learning*: Traj2Vec encodes each transformed ST-Seq (with road segment ID replaced by embeddings) as a trajectory embedding, that captures inherent spatial and temporal properties of trajectories and the underlying road network, for use in various trajectory mining applications. To the best of our knowledge, this is the first attempt to seamlessly exploit both moving behaviors of mobile users and road networks for trajectory representation learning.

There are only a few prior studies on trajectory representation learning [18, 44]. Although these prior works all claim that their approaches are able to capture the moving behaviors in trajectories by their proposed RNN-based autoencoder models, they fail to capture both the spatial and temporal properties of trajectories, and miss the topological constraints of the underlying road network. Among them, by defining several manual-crafted feature functions to aggregate sample points and applying the seq2seq model to learn trajectory embeddings, trajectory2vec [44] does not consider the geometric locations of the sample points in trajectories and thus fail to capture the spatial property of trajectories. However, by partitioning the space into cells, t2vec [18] transforms a trajectory as a sequence of cells and then designs an RNN-based model to encode a transformed trajectory into a latent vector. Without considering the time information of the sample points in trajectories, it fails to capture the temporal properties of trajectories. Moreover, these previous works do not consider the underlying road networks even though the trajectories are capturing movement of mobile road users. Finally, both works only focus on one specific application and thus do not meet the ultimate goal of representation learning, i.e., generally supporting various applications.

The main contributions of this study are threefold:

- **Novel ideas for trajectory representation learning by exploiting the moving behaviors of mobile road users and properties of road networks**. This article analyzes the challenges of trajectory representation learning problem and proposes innovative ideas to encode the inherent spatial and temporal properties of trajectories into trajectory embeddings by exploiting the underlying road networks to overcome these challenges.
- **A new representation learning framework for trajectories**. We propose a three-phase framework, *Trembr*, to learn representations of trajectories. The novelty of this framework lies in the design of a neural network–based model, *Road2Vec*, to learn road segment representations that captures various relationships amongst road segments and the design of a novel RNN-based encoder–decoder model, *Traj2Vec*, for trajectory representation learning to capture the spatial and temporal properties inherent in trajectories while constraining the learning process upon the topological structure of the road network.
- **Comprehensive empirical evaluation using real-world data**. We evaluate Trembr by conducting a comprehensive evaluation with three different tasks, trajectory similarity measure, travel time prediction, and destination prediction, using two real-world trajectory datasets. Compared with baselines (seq2seq) and two state-of-the-art representation learning models (trajectory2vec and t2vec), empirical result shows that Trembr soundly outperforms all existing models by at least one order of magnitude in terms of mean rank in trajectory similarity measure, 23.3% to 41.7% and 39.6% to 52.4% in terms of MAE in travel time prediction and destination prediction, respectively.

In the rest of this article, we first review the related work in Section 2, and provide research background, problem definition and analysis in Section 3. We present the proposed Trembr framework in Section 4 and show experiment results in Section 5. Finally, we conclude the article in Section 6 and discuss future research directions.

2 RELATED WORK

We briefly review the related work on trajectory representation learning, network representation learning, and road network matching of trajectories.

2.1 Representation Learning of Trajectories

The goal of representation learning is to automatically transform raw data into general-purpose low-dimensional latent vectors that effectively serve as input features to machine learning and data mining algorithms for various applications [2]. In the past several years, research interests in neural network–based representation learning algorithms have grown rapidly in various domains, including text processing [16, 25, 30], network analytics [31, 39], computer vision [33, 34], and so on. For sequence data, e.g., text, videos and audios, RNN-based encoder–decoders, e.g., RNN autoencoder, sequence to sequence (seq2seq), skip-thought vectors, and attention models [5, 7, 15, 19, 37], have been developed to capture the sequential order in those data.

Research on representation learning of trajectories, which can be considered as a kind of sequence data has been reported in the literature only recently. As mentioned previously, it is natural to consider RNN-based encoder–decoders to learn representations for sequence data. However, traditional RNN-based encoder–decoders are designed for textual data in natural language processing, where a textual document seldom has noises (e.g., typos) and does not have time gaps between words. Therefore, simply applying existing RNN-based encoder–decoders for trajectory representation learning is impractical, because existing RNN-based encoder–decoders are not designed to handle the non-uniform and low sampling rate and noisy sample points in raw trajectory data. Moreover, they fail to consider the spatial and temporal properties inherent in trajectories and ignore the physical constraints imposed by road networks upon behaviors of mobile road users, e.g., vehicles.

To the best knowledge of the authors, there are only two prior studies on representation learning of trajectories [18, 44]. Although these prior works all claim successes in capturing the moving behaviors in trajectories, they do not handle well the various issues discussed earlier. Between them, *trajectory2vec* [44] employs a sliding window over trajectories to extract several manual-crafted moving features from the raw sample points in the window, e.g., time interval, moving distance, moving speed and rate of turns. Then it simply applies the seq2seq model to encode a trajectory (in form of a sequence of the moving features) into a latent vector. However, without considering the geographic locations of sample points in trajectories, trajectory2vec fails to capture the spatial properties of trajectories. As a result, trajectories may have similar representations even if they are generated by different routes taken by different mobile users. Adopting a different approach to handle the issues of noisy sample points and low sampling rate, t2vec [18] partitions the space in grid to transform a trajectory as a sequence of grid cells. Then, it designs an RNN-based encoder–decoder with a spatial-aware loss function to encode a trajectory (in form of a sequence of cells) into a latent vector. The spatial-aware loss function penalizes the misses in prediction during decoding by considering the geographic distance between cells. However, without considering the time information of the sample points in trajectories, t2vec fails to capture the temporal properties of trajectories. For both of these existing works, selecting a proper size of the sliding window or the cells is critical. Too large a size incorporates irrelevant points into embeddings, while too small a size leads the learning process to suffer the data sparsity issue. Moreover, both works only consider one application in their evaluations, without demonstrating the generality and applicability of their trajectory embeddings for multiple applications. Finally, these previous works do not consider road maps in their representation learning, even though most of the trajectory datasets are capturing the movement of mobile users on roads.

2.2 Representation Learning in Networks

Recently, research on representation learning has been extended to network data, which aims to embed a network into a low-dimensional space and represent each node or edge as a low-dimensional feature vector for further applications. However, instead of the road networks targeted in this work, most existing works focus only on learning node or edge vectors in general networks, such as homogeneous information networks [12, 31, 39], heterogeneous information networks [8, 9], attributed network [14, 17], and so on. Simply applying the existing network representation learning methods for road network representation learning is impractical. First of all, they do not consider the spatial properties of road networks, e.g., the lengths of road segments. Second, network representation learning methods usually apply random walks to sample training data, which fails to capture the user movement behaviors in road networks, i.e., the trajectories of mobile users. Finally, network representation learning methods usually do not consider to incorporate various of relationships among nodes or edges, such as the same road types and frequent co-occurrence in trajectories.

2.3 Road Network Matching

Raw trajectories, typically suffering from low and non-uniform sampling rates and noisy sample points, may not perfectly match the underlying road networks the mobile users traveling on. To address these issues, existing noise filtering methods fall into several major categories, such as mean filtering, heuristics-based outlier detection, road network matching, and so on [45]. Among them, road network matching, i.e., aligning the sample points of a trajectory onto the road network to transform the trajectory as a sequence of road segments, is an essential step in data preprocessing for location-based services [22], which not only helps to filter noisy data but also discover the exact path of the trajectory on the road network. Several road network matching works have been developed over the years [1, 3, 11, 22, 27, 42]. These works can be classified as geometric or topological techniques [26]. Geometric techniques utilize the geometry of a trajectory and the road network for matching, e.g., point-to-curve matching [42]. They only consider the shapes of the road segments in the road network regardless of their topological connectivity. Therefore, these techniques fail to achieve good matching accuracy for trajectories that suffer low and non-uniform sampling rates and noisy sample points, because the geometry of trajectories got distorted. However, topological techniques, e.g., References [1, 3, 11, 22, 27] leverage the connectivity of road segments for road network matching. Among those, Hidden Markov Model–based approaches [11, 22, 27] provide probabilistic frameworks to address the noises in the input trajectory with remarkable accuracy. In this article, we adopt *Barefoot*, a state-of-the-art Hidden Markov Model–based model proposed by Newson and Krumm [24, 27], for road network matching. Barefoot aims to find the most likely the sequence of road segments in a road network represented by a trajectory. More specifically, in Barefoot, individual road segment are modeled as the states of the HMM model, and the goal is to match each GPS sample point to the proper road segments while exploiting the connectivity of the road network to find the most like transitions between road segments.

3 PRELIMINARIES

In this section, we first introduce the notions of trajectory, road network, and ST-Seq, present the general framework of RNN-based encoder–decoders, and then define the targeted research problem and discuss the challenges.

3.1 Trajectory, Road Network, and ST-Seq

We first define trajectories and road networks.

Definition 1 (Trajectory). A trajectory T is a sequence of spatio-temporal sample points generated from the movement of a mobile user. A sample point p contains a location (x, y) (i.e., longitude and latitude) and a timestamp t.

Figure 4 shows a part of a trajectory T_1 extracted from a real dataset of taxi trajectories collected in the city of Proto [4]. As shown, many sample points are not exactly located on the roads due to noises or low sampling rate.

Definition 2 (Road Network). A road network is a directed graph $G = (V, E, \Psi)$, where V is a set of nodes (i.e., intersections); $E \subseteq V \times V$ is a set of directed edges that denote road segments; and $\Psi : E \rightarrow R$ is a type mapping function of edges. Each edge $e \in E$ corresponds to a road segment from a start node $e.s = v \in V$ to an end node $e.e = v' \in V$, where $v' \neq v$. Moreover, an edge $e \in E$ belongs to a particular edge type in R, i.e., $\Psi(e) \in R$.

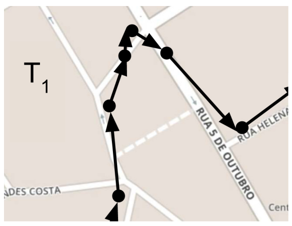


Fig. 4. A trajectory.

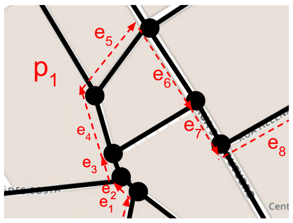


Fig. 5. An ST-Seq.

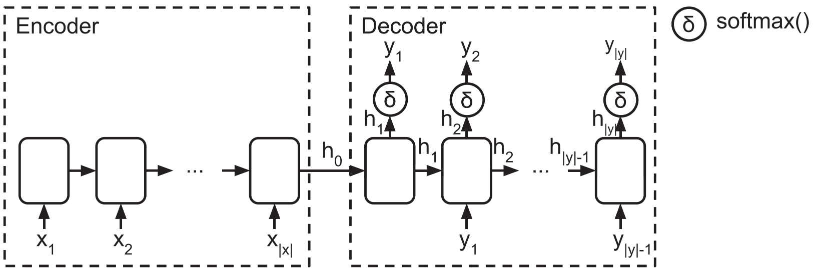


Fig. 6. The RNN-based encoder–decoder model.

Definition 3 (Spatial-Temporal Sequence (ST-Seq)). An ST-Seq $p = \{(r_1, t_1), \ldots, (r_{|p|}, t_{|p|})\}$ is a sequence of road segments and travel times, where $(r_i,\ t_i)$ denotes the ith road segment and the corresponding travel time.

Figure 5 shows an example of the road network corresponding to the road map shown in Figure 4. Here the trajectory T_1 in Figure 4 is expressed as an ST-Seq $p_1 = \{(e_1, 5sec), (e_2, 2sec), (e_3, 5sec), \ldots, (e_8, 10sec)\}$ (as illustrated by dotted arrows).

3.2 RNN-based Encoder–Decoders

We briefly present the general framework of RNN-based encoder–decoders (as illustrated in Figure 6). Consider a pair of sequences $(x = \{x_1, \ldots, x_{|x|}\}, y = \{y_1, \ldots, y_{|y|}\})$, where x_i and y_i denote the ith token of x and y (e.g., a word in a document or a signal in a speech), and $|x|$ and $|y|$ denote the lengths of x and y. An RNN-based encoder–decoder model, e.g., RNN autoencoder [5] or the sequence to sequence model [37], aims to encode x into a low-dimensional latent representation h_0, which is in turn decoded back to y by maximizing the conditional probability $P(y|x)$. As such, h_0 preserves the sequential information in x.

The conditional probability $P(y|x)$ is modeled below.

$$P(y|x) = P(y_1, \ldots, y_{|y|}|x) = P(y_1|x) \prod_{i=2}^{|y|} P(y_i|y_{1:i-1}, x),$$

where $y_{1:i-1}$ denotes the subsequence $y_1, y_2, \ldots, y_{i-1}$. Since the encoder encodes the sequential information in x into h_0, the decoder derives $P(y_1|x)$ and $P(y_i|y_{1:i-1}, x)$ as follows:

$$P(y_1|x) = P(y_1|h_0)$$
$$P(y_i|y_{1:i-1}, x) = P(y_i|y_{1:i-1}, h_0).$$

The decoder sequentially computes $P(y_i|y_{1:i-1}, h_0)$ at each position i of y. Specifically, at position i, the decoder transforms $y_{1:i-1}$ and h_0 into the hidden state h_i, which preserves the sequential information of x and $y_{1:i-1}$ and then predicts y_i by h_i. h_i is computed from the previous token

y_{i-1} and the output of previous position h_{i-1} by $h_i = f(y_{i-1}, h_{i-1})$, where $f(\cdot, \cdot)$ is the activation function of an RNN cell, e.g., long short-term memory (LSTM) [13] or gated recurrent unit (GRU) [6]. After computing h_i at position i in y, $P(y_i|y_{1:i-1}, h_0)$ can be derived as follows:

$$P(y_i = u|y_{1:i-1}, h_0) = P(y_i = u|h_i) = \frac{exp(W_u \cdot h_i)}{\sum_{v \in N} exp(W_v \cdot h_i)},$$

where W is the projection matrix that projects h_i from the hidden state space into the vocabulary space, W_u denotes the uth row of W, and N is the vocabulary.

3.3 Problem Statement

This work aims to learn low-dimensional embeddings for trajectories in a given trajectory dataset by exploiting the underlying road network. In the following, we formally state our research goal.

Definition 4 (Trajectory Representation Learning with Road Networks). Given a trajectory dataset D and an underlying road network G, develop an end-to-end framework that learns a function $f : D \rightarrow \mathbb{R}^d$ to project each trajectory $T \in D$ as a vector in a d-dimensional space $\mathbb{R}^d$ in support of a variety of trajectory mining tasks.

Ideally, the learned embedding of a trajectory shall reflect the inherent characteristics and underlying route of the trajectory on the road network. Meanwhile, the similarity of two trajectories based on the learned embeddings shall be robust to non-uniform, low sampling rates and noisy sample points.

3.4 Our Ideas and Faced Challenges

In this article, we propose a framework, *Trembr*, to tackle the problem of trajectory representation learning by exploring underlying road networks. Based on Trembr, we first match each trajectory onto the road network and transform a trajectory as an ST-Seq aiming to suppress the impact of noisy and missing sample points caused by low and non-uniform sampling rates. Next, we propose a novel neural network model, *Road2Vec*, to learn representations of road segments by capturing explicit and implicit relationships between road segments for further trajectory representation learning. Finally, we propose a new RNN-based encoder–decoder, *Traj2Vec*, to learn representations of trajectories by capturing the spatial and temporal properties inherent in trajectories. To implement the Trembr framework, we face new challenges: (1) Road segment representation learning. A well-designed neural network model is critical for effective and efficient learning to capture multiple relationships between road segments, such as frequent co-occurrence in trajectories and the same road types. Moreover, how to define the co-occurrence relationship between road segments? How to prepare training data for the different types of relationships between road segments? (2) Trajectory representation learning. A well-designed RNN-based auto-encoder is critical for effective and efficient learning to encode both the spatial and temporal properties of trajectories into embeddings. How to incorporate the topology of the underlying road network to constraint the learning? What is the proper loss function for model learning? These are research questions arising in the design of Trembr.

4 THE TREMBR FRAMEWORK

As introduced earlier in Figure 3, the Trembr framework consists of three phases: *Road network matching*, *Road segment representation learning*, and *Trajectory representation learning*. In the section, we detail our design in each phase.

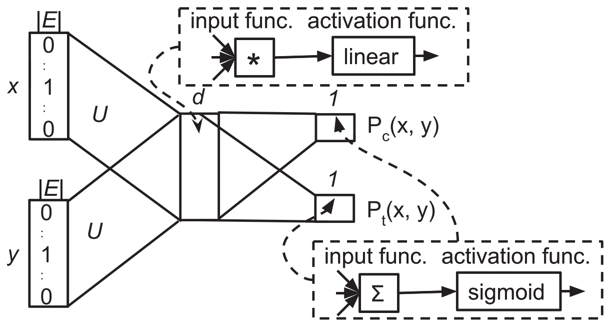


Fig. 7. The Road2Vec model.

4.1 Road Network Matching

Raw trajectory data typically suffers from low and non-uniform sampling rates and noisy sample points, due to sensor noise or poor positioning signals. As existing RNN models are not designed to handle these issues, it is not a good idea to directly use them for trajectory representation learning. Moreover, they are not able to capture the characteristics and constraints of physical paths underlying the trajectories. Therefore, we preprocess the trajectories to handle low and non-uniform sampling rates and filter noises before starting the learning process.

As mentioned, we argue that road network matching methods, projecting a trajectory onto a road network and transform it as a sequence of road segments, may not only help to filter noisy data but also discover the exact path of the trajectory on the road network (i.e., handle the low and non-uniform sample rate issue), which along with the topological structure of the road network are beneficial to trajectory representation learning. Thus, in Trembr, we adopt *Barefoot*, a Hidden Markov Model–based model [24, 27], for road network matching. More specifically, after matching a trajectory on a road network, each GPS sample point is re-located on a road segment. We simply assume that the moving speed between two consecutive sample points (there may be multiple road segments between them) is constant. Based on this assumption, a trajectory dataset D is transformed into an ST-seq dataset D_p as defined in Section 3.1.

4.2 Road Segment Representation Learning

Conventionally, road segments are identified by unique IDs. As argued previously, road segments are not independent from each other but relevant in terms of some relationships, such as the same road types (an explicit relationship in a road network) and frequent co-occurrence in trajectories (an implicit relationship in moving behaviors). Simply using a road segment's own attributes, e.g., unique IDs or geo-locations of its source/destination nodes, to represent the road segment fails to capture those informative relationships. To address this issue, we propose Road2Vec, a new neural network model designed by exploring the aforementioned relationships, to learn road segment embeddings as part of ST-Seqs for use in trajectory representation learning. Specifically, Road2Vec learns an embedding for each road segment by jointly predicting whether two road segments co-occur in the same trajectory and whether two road segments belong to the same road type. In the following, we first present the Road2Vec model and the optimization process, and then introduce the training data preparation for Road2Vec and some related issues.

4.2.1 The Road2Vec Model. As shown in Figure 7, the Road2Vec model is a *multi-task binary classifier* that takes a pair of road segments x and y (encoded as *one-hot* vectors) as the inputs to predict the aforementioned two relationships, correspondingly measured by (a) the probability

$P_c(x,y)$ for x and y to co-occur in a trajectory, and (b) the probability $P_t(x,y)$ for x and y to have the same road type. The two joint probabilities are derived below.

$$P_c(x,y) = \sigma(U_x \cdot U_y) \qquad\qquad P_t(x,y) = \sigma(U_x \cdot U_y),$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the Sigmoid function, U is a matrix consisting of all road segments' embeddings, and U_x is the xth row of U, denoting the embedding for road segment x. In the training process, if road segments x and y are observed in the training data to satisfy one of the targeted prediction tasks, U_x and U_y are moved close in the latent space. Otherwise, they are moved away.

4.2.2 *Optimization Objective.* To train Road2Vec, a training set D_r containing training data in the form of $d = \langle x, y, L_c(x,y), L_t(x,y)\rangle$ is extracted from an ST-Seq dataset D_p (the preparation of training data to be detailed later). Here $L_c(x,y)$ and $L_t(x,y)$, two boolean values, indicate whether x and y co-occur in a trajectory and whether x and y belong to the same road type, respectively. With D_r, Road2Vec is trained by the backpropagation training algorithm in conjunction with stochastic gradient descent. It goes backwards to adjust the weights in U for each entry in D_r, attempting to maximize the objective O, which is the combination of $O_{c_d}(x,y)$ and $O_{t_d}(x,y)$ of for each entry in D_r. Here $O_{c_d}(x,y)$ and $O_{t_d}(x,y)$ quantify how Road2Vec correctly predicts $L_c(x,y)$ and $L_t(x,y)$ for a data entry d, respectively. In specific, given a training data entry $d = \langle x, y, L_c(x,y), L_t(x,y)\rangle$, $O_{c_d}(x,y)$ aims to maximize $P_c(x,y)$, when $L_c(x,y)$ is 1; and minimize $P_c(x,y)$, otherwise. However, $O_{t_d}(x,y)$ aims to maximize $P_t(x,y)$, when $L_t(x,y)$ is 1, and minimize $P_t(x,y)$, otherwise. To ease the computation in the optimization process, we maximize $\log O_{c_d}(x,y)$ and $\log O_{t_d}(x,y)$ rather than directly maximize $O_{c_d}(x,y)$ and $O_{t_d}(x,y)$, which are derived as follows:

$$O_{c_d}(x,y) = \begin{cases} P_c(x,y), & \text{if } L_c(x,y) = 1 \\ 1 - P_c(x,y), & \text{if } L_c(x,y) = 0 \end{cases}$$

$$\log O_{c_d}(x,y) = L_c(x,y)\log P_c(x,y) + [1 - L_c(x,y)]\log[1 - P_c(x,y)]$$

$$O_{t_d}(x,y) = \begin{cases} P_t(x,y), & \text{if } L_t(x,y) = 1 \\ 1 - P_t(x,y), & \text{if } L_t(x,y) = 0 \end{cases}$$

$$\log O_{t_d}(x,y) = L_t(x,y)\log P_t(x,y) + [1 - L_t(x,y)]\log[1 - P_t(x,y)].$$

The overall objective function O is defined as follows:

$$O = \sum_{d \in D_r} \left\{\alpha \log O_{c_d}(x,y) + (1-\alpha)\log O_{t_d}(x,y)\right\},$$

where α is for weighing $O_{c_d}(x,y)$ and $O_{t_d}(x,y)$.

We then apply the stochastic gradient descent algorithm to maximize the objective function O. Specifically, for each training data entry, $d = \langle x, y, L_c(x,y), L_t(x,y)\rangle$, it goes backwards to adjust the weights of road segments x and y in U based on the gradients, as shown:

$$U_x := U_x + \frac{\alpha d \log O_{c_d}(x,y) + (1-\alpha) d \log O_{t_d}(x,y)}{dU_x}$$
$$U_y := U_y + \frac{\alpha d \log O_{c_d}(x,y) + (1-\alpha) d \log O_{t_d}(x,y)}{dU_y}$$

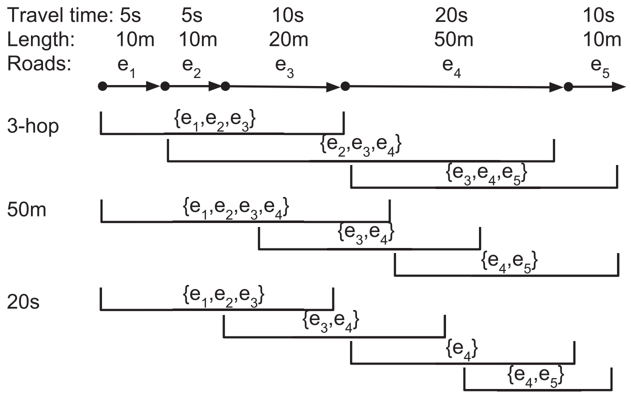


Fig. 8. Different sliding window sizes.

4.2.3 Training Data Preparation. To prepare training data for Road2Vec, we use the ST-Seq dataset D_p processed in Phase 1 to generate the training data in the form of $d = \langle x, y, L_c(x, y), L_t(x, y)\rangle$. Given an ST-Seq, we apply a sliding window with a certain window size (to be discussed later) over the ST-Seq to extract the co-occurrence relationship of road segments in the ST-Seq. There are various ways to define the window size, e.g., within-k-hop, moving distance, or moving time. Consider an example ST-Seq $p = \{(e_1, 5\,sec), (e_2, 5\,sec), (e_3, 10\,sec), (e_4, 20\,sec), (e_5, 10\,sec)\}$, as shown in Figure 8. Suppose we use a sliding window of *within-3-hop* window size and 1-hop sliding step. Three windows of road segments are extracted from p, i.e., $\{e_1, e_2, e_3\}$, $\{e_2, e_3, e_4\}$, and $\{e_3, e_4, e_5\}$ as shown in Figure 8 (the "3-hop" rows). For each pairs of road segments x and y within a window, we create a positive training entry reflecting their co-occurrence relationship in a trajectory. We also check whether x and y have the same type to generate training data entries for the same-type relationship. For example, in the first window $\{e_1, e_2, e_3\}$, we generate positive training data entries $\langle e_1, e_2, 1, same(e_1, e_2)\rangle$, $\langle e_1, e_3, 1, same(e_1, e_3)\rangle$, and $\langle e_2, e_3, 1, same(e_2, e_3)\rangle$, and so on, where $same(x, y)$ is filled by checking whether x and y have the same road type. Notice that, within-k-hop, without considering the moving distance and the moving time in the ST-Seq, may potentially group irrelevant road segments together or miss relevant road segments. For example, in Figure 8, e_3 and e_5 are distant but grouped by a within-3-hop window. As a result, we also explore two alternative definitions of window size, *moving distance* and *moving time*, which reflect the lengths of road segments and the moving behavior of mobile users, respectively. Figure 8 also shows the road segment groups by using 50-m moving distance (with a 25-m sliding step) and using 20-s moving time (with 10-s sliding step) as the window sizes as shown in Figure 8 (the "50 m" and "20 sec" rows). While generating positive samples by sliding window over trajectories, we also prepare negative data entries following the ideas of *negative sampling* [25]. For each sampled positive entry, we generate negative training data entries by randomly replacing one of the two values with either x' or y', where x' and y' are randomly selected road segments.

4.3 Trajectory Representation Learning

We argue that existing RNN-based encoder–decoders fail to capture the spatial and temporal properties inherent in trajectories and ignore the physical constraints imposed by road networks upon user movements. To address these issues, Traj2Vec takes an ST-Seq (with road segment IDs replaced by Road2Vec embeddings) as input to generate a low-dimensional trajectory embedding,

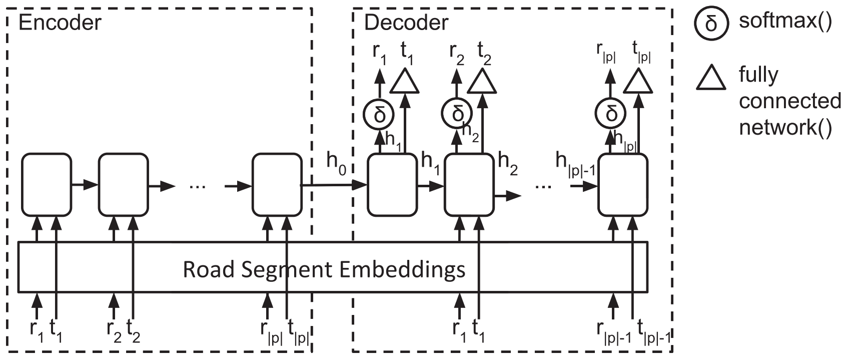


Fig. 9. The Traj2Vec model.

by capturing both the spatial and temporal properties in the trajectory. Further, we embed the road network topological constraint in the decoding process and propose a novel *road segment relevance-aware loss function* for model learning. In the following, we first present the details of the Traj2Vec model and the loss function, and then discuss issues arising in Traj2Vec.

4.3.1 The Traj2Vec Model. As shown in Figure 9, the Traj2Vec model is a *multi-task* RNN-based encoder–decoder that takes an ST-Seq $p = \{(r_1, t_1), \ldots, (r_{|p|}, t_{|p|})\}$ as the input. The proposed model first sequentially encodes the input p into a low-dimensional latent vector h_0 (i.e., the trajectory embedding) and in turn decodes the embedding h_0 back to p by jointly maximizing the probability for a road segment r_i to be traveled following the partial path $p_{1:i-1}$ and minimizing the error in predicting the travel time t_i at each state i to capture both the spatial and temporal properties in the trajectory. More specifically, given an ST-Seq p, during the encoding process, the model sequentially takes pairs (r_i, t_i) of p $(i = 1, ..|p|)$ as the input for their corresponding states. At the ith state, r_i is replaced by its embedding (learned in Phase 2) and is concatenated with t_i (a positive float-point value) to serve as the input vector. After encoding, Traj2Vec takes h_0, the output of the $|p|$th state in the encoding process, as the input to the decoder that sequentially decodes the intermediately generated latent vector $h_{i-1}(i = 1..|p|)$ back to p by predicting each road segment r_i and its corresponding travel time t_i. Here we apply two-layer LSTM cell in the model for both encoding and decoding. As mentioned previously, the road segment transitions in a trajectory are strictly constrained by the topology of the road network. However, it is proved that RNN models hardly learn the topology information well automatically [43]. Therefore, we embed the topological constraint in the decoding process by deriving $P(r_i|p_{1:i-1}, h_0)$, i.e., the conditional probability for a road segment r_i to be traveled on, given that the previous $i-1$ segments have been passed by, as follows:

$$P(r_i = e_u|p_{1:i-1}, h_0) = P(r_i = e_u|h_i) = \frac{exp(W_{r_u} \cdot h_i)}{\sum_{e_v \in E} exp(W_{r_v} \cdot h_i \cdot l_{e_u,e_v})}$$
$$\text{where } l_{e_u,e_v} = \begin{cases} 1, & e_u.s = e_v.s \\ 0, & otherwise \end{cases}.$$

Here W_r is the projection matrix that projects h_i from the hidden state space into the space of all distinct road segments, W_{r_u} denotes the uth row of W_r, and l_{e_u,e_v} is the embedded topological constraint, which indicates whether e_u and e_v follow the same road segment r_{i-1} in the road network. Constraining the transition probabilities of road segments not following r_{i-1} to zero allows the model to focus on updating only the weights of those road segments following r_{i-1}. Moreover,

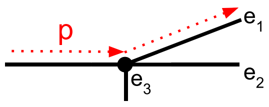


Fig. 10. Relevance between road segments.

the prediction of the travel time t_i for road segment r_i, denoted by $f_{time}(p_{1:i-1}, h_0)$, is derived as follows:

$$f_{time}(p_{1:i-1}, h_0) = f_{time}(h_i) = FFN(h_i),$$

where $FFN(x) = \left(W_{t2} \cdot (W_{t1} \cdot x + b_1) + b_2\right)$ is a simple fully connected feed-forward network for travel time prediction as a regression task, which consists of one hidden layer with linear transformations. In $FFN(x)$, W_{t1} is a $(\frac{d}{2} \times d)$ matrix, W_{t2} is a $(1 \times \frac{d}{2})$ matrix and d is the dimensionality of h_0.

4.3.2 Optimization Objective. Using a training set D_p containing all ST-Seqs, in the form of $p = \{(r_0, t_0), \ldots, (r_{|p|}, t_{|p|})\}$, Traj2Vec is trained by the backpropagation training algorithm in conjunction with gradient descent. It goes backwards to adjust the parameters of Traj2Vec for each entry in D_p, attempting to minimize the loss L, which is a combination of $L_{road_p}(p)$ and $L_{time_p}(p)$ for each training data entry p in D_p, where $L_{road_p}(p)$ and $L_{time_p}(p)$ are the losses for road segments decoding and travel time decoding, respectively. To ease the computation in the optimization process, we minimize $\log L_{road_p}(p)$ and $\log L_{time_p}(p)$ instead of $L_{road_p}(p)$ and $L_{time_p}(p)$. In the following, we first derive $\log L_{road_p}(p)$ and $\log L_{time_p}(p)$ based on $P(r_i|p_{1:i-1}, h_0)$ (the conditional probability for traveling on road segment r_i) and $f_{time}(p_{1:i-1}, h_0)$ (the prediction of its travel time t_i),

$$\log L_{road_p}(p) = -\log \prod_{i=1}^{|p|} P(r_i = e_u|p_{1:i-1}, h_0) = -\sum_{i=1}^{|p|} \log \frac{exp(W_{r_u} \cdot h_i)}{\sum_{e_v \in E} exp(W_{r_v} \cdot h_i \cdot l_{e_u,e_v})}$$

$$\log L_{time_p}(p) = 2 \sum_{i=1}^{|p|} \log\{t_i - f_{time}(p_{1:i-1}, h_0)\}.$$

Note that $L_{road_p}(p)$ above does not capture the relevance between road segments very well. Actually, $L_{road_p}(p)$ penalizes the misses in predicting the output road segments with equal weights, without considering the relevance between a predicted road segment and the target road segment (the observed road segment in the trajectory). Intuitively, a predicted road segment that is more relevant to the target road segment, e.g., in terms of their Euclidean distance or cosine similarity of their road segment embeddings, is more accurate than those that are less relevant. For example, in Figure 10, if the target road segment is e_1, the loss function penalizes the predictions e_2 and e_3 *equally* (which is not a good penalty scheme as e_2 is more relevant to e_1 than e_3 to e_1). In this case, it is more acceptable for the decoder to output e_2 instead of e_3. We address this issue by proposing a road segment relevance-aware loss function to assign a weight for each road segment e_i in decoding of a target road segment e_t. Our idea is to assign the weight of road segment e_i proportional to the relevance (e.g., measured by Euclidean distance or cosine similarity between e_i and e_t's embeddings). As such, the more relevant road segments to e_t have the less weights. The road segment relevance-aware loss function $\log L'_{road_p}(p)$ (based on Euclidean distance) is given

Table 1. Statistics of Trajectory Datasets

Name	#Trajectories	Avg. #Points	Avg. time gap
Porto	1,233,766	60.20	15.00 sec.
Tokyo	273,046	31.76	14.37 sec.

as:

$$\begin{aligned}\log L'_{road_p}(p) &= -\sum_{i=1}^{|p|} \log \frac{exp(W_{r_u} \cdot h_i)}{\sum_{e_v \in E} exp(W_{r_v} \cdot h_i \cdot l_{e_u,e_v} \cdot w_{e_u,e_v})} \\ &= -\sum_{i=1}^{|p|} \left\{ W_{r_u} \cdot h_i - \log \sum_{e_v \in E} exp(W_{r_v} \cdot h_i \cdot l_{e_u,e_v} \cdot w_{e_u,e_v}) \right\},\end{aligned}$$

where $w_{e_u,e_v} = \frac{|vec(e_u) - vec(e_v)|_2}{\gamma}$ is the weight for e_v when the target for decoding is e_u, $|vec(e_u) - vec(e_v)|_2$ denotes the Euclidean distance between the embeddings of e_u and e_v and γ is a road similarity scale parameter. Finally, the loss function L is derived as follows:

$$L = \sum_{p \in D_p} \left\{ \beta \log L'_{road_p}(p) + (1 - \beta) \log L_{time_p}(p) \right\},$$

where β is for weighing $\log L'_{road_p}(p)$ and $\log L_{time_p}(p)$.

4.4 Fine-tuning Road Segment Embeddings

Based on the idea of *fine-tuning* in transfer learning [41], the road segment embeddings (learned in Phase 2) not only can be precomputed for replacing each r_i (as a fixed feature vector) in an ST-Seq for trajectory representation learning. They can also be further fine-tuned through the process of trajectory representation learning to obtain better trajectory embeddings. More specifically, the precomputed road segment embeddings obtained in Phase 2 can be used as the initial latent vectors for r_i's of ST-Seqs in Phase 3. Then, while learning the trajectory embeddings, the latent vector of each r_i is further updated to optimize the loss function of trajectory representation learning. We show later that employing this fine-tuning in Phase 3 slightly improves the performance.

5 EXPERIMENTS

In this section, we conduct an empirical evaluation on Trembr using two real-world trajectory datasets. The experimentation also examines several models for trajectory representation learning, including two baselines and two existing works for comparison. To demonstrate the generality of embeddings learned by Trembr and other models, *trajectory similarity measure*, *travel time prediction*, and *destination prediction*, are adopted as benchmark tasks.

5.1 Datasets, Models, and Evaluation Platform

The following describes the real-world trajectory datasets used in the evaluation. Some statistics of the trajectories extracted from these datasets are summarized in Table 1.

Porto taxi data, made available for the Taxi Service Trajectory Prediction Challenge@ ECML/PKDD 2015 [4], contains 1.7 million taxi trajectories of 442 taxis running in Porto, Portugal over 19 months. Each taxi reports its location every 15 s. We remove trajectories with less than 10 GPS sample points, which yields 1.23 million trajectories.

Tokyo OpenPFLOW data have collected 78 million GPS sample points from 617,040 users in Tokyo walking or taking different vehicles, such as bike, car, and train, for 1 day [38]. We extract the sequences of GPS points by users taking bike or car and then segment them into trajectories based on a 45-s time gap. Finally, we remove trajectories with less than 10 GPS sample points to yield 0.27 million trajectories.

The trajectory representation learning models evaluated for comparison includes two baselines and two state-of-the-art methods. We also investigate different ways of exploiting Trembr to generate embeddings, i.e., by feeding trajectories in forward or backward (reversed order) directions and their combinations. The compared models are summarized below.

Raw Trajectory uses raw trajectories (sequences of GPS samples without being transformed as road segment sequences) as inputs for the seq2seq model.

Road Segments uses transformed sequences of road segments (no embeddings) as inputs for the seq2seq model.

trajectory2vec [44] transforms trajectories as sequences of aggregated moving features as inputs for the seq2seq model.

t2vec [18] transforms trajectories as sequences of cells on the map as inputs for the t2vec model (which is an RNN-based encoder–decoder with spatial-aware loss function).

Trembr$_f$ and **Trembr$_b$**. ST-Seqs (with road segment embeddings) feed in forward and backward directions as inputs for the proposed Trembr framework, denoted as Trembr$_f$ and Trembr$_b$, respectively.[2]

Trembr$_{f+b}$. A hybrid of Trembr$_f$ and Trembr$_b$ by concatenating the ($d/2$-dimensional) embeddings learned by Trembr$_f$ and Trembr$_b$ into a d-dimensional embedding.

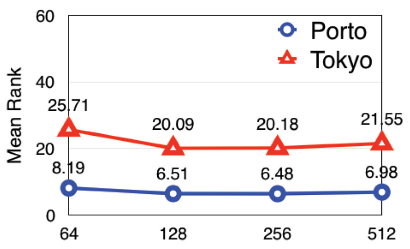
Trembr, implemented in C (Road2Vec) and python with Tensorflow (Traj2Vec), is trained using a GeForce GTX 1080 GPU. All experiments are run on the Ubuntu 18.04 operating system with an Intel Core i5-8400 CPU.

5.2 Trajectory Similarity Measure

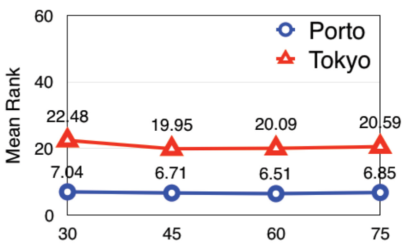
In this section, we evaluate the models by the task of using embeddings to measure trajectory similarity. We first introduce the experimental setup, including the evaluation methodology for trajectory similarity, the preparation of datasets, and the default settings of parameters in the compared models. Then, we perform sensitivity tests on parameters of Trembr to determine the default settings. Next, we examine several issues in the Trembr framework, including the window size in Road2Vec, embedding network topology in Traj2Vec and road segment relevance in the loss function in Traj2Vec. Finally, we show the experimental results and some case study.

5.2.1 Experimental Setup. The measure of trajectory similarity is essential in many trajectory mining tasks and applications. Due to the lack of ground truth for evaluation of trajectory similarity, two recent studies [32, 36] propose to exploit the idea of *self-similarity* that exists in sub-trajectories of the same trajectory for empirical evaluation. Inspired by this idea and the experiment adopted in Reference [18], we design an experiment to evaluate the effectiveness of using trajectory embeddings (generated by various models) for similarity measure.[3] In specific, for each trajectory T in a given trajectory dataset D, we take odd-numbered and even-numbered GPS sample points from T to create two interleaving sub-trajectories, T_a and T_b. Thus, we have two datasets, $D_a = T_a$ and $D_b = T_b$. Next, we randomly choose 10,000 trajectories from D_a and their corresponding trajectories in D_b to form test datasets, denoted as Q_a and Q_b, respectively. We use

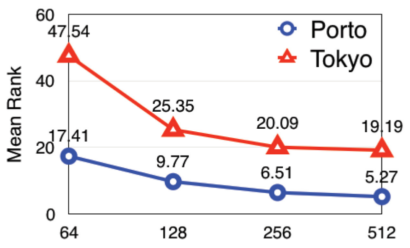
[2]We have actually tried other alternative ways to apply Trembr, e.g., using forward trajectories for encoding and backward trajectories for decoding in Traj2Vec, but Trembr$_f$ and Trembr$_b$ perform better.
[3]Here the trajectory similarity is calculated by Euclidean distance between embeddings of two trajectories.



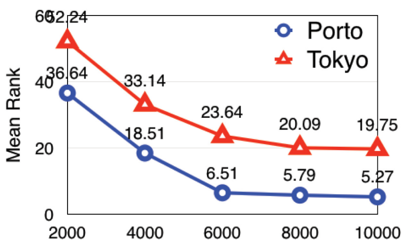
(a) Dimensionality in Road2Vec



(b) Window size in Road2Vec



(c) Dimensionality in Traj2Vec



(d) Number of Iterations in Traj2Vec

Fig. 11. Parameter sensitivity – Trembr$_{f+b}$.

D_a to train the models that in turn are used to infer the embeddings of all the trajectories in Q_a and Q_b. Finally, for each trajectory $T_a \in Q_a$, we rank trajectories in Q_b in terms of their Euclidean distance to T_a in the space of trajectory embeddings. Ideally, the corresponding T_b shall be ranked at the top, since T_a and T_b are created from the same original trajectory. We use *mean rank* as the metric for the task of trajectory similarity measure.

Regarding default parameter settings, the dimensionality of trajectory embeddings for all models is set to 256, which usually achieves the best or converged results. For Trembr, in Road2Vec, the dimensionality of road segment embeddings are set to 128. The window size adopts `Moving time` with 60 s, the number of negative samples per positive sample is set to 5 and the α for weighting the loss is set to 0.9. In Traj2Vec, the β for weighting the loss is set to 0.9, the maximum length of input ST-Seqs for training is determined by data while it can cover more than 90% ST-Seqs, (e.g., 150 for the Porto dataset and 100 for the Tokyo dataset) and the road segment embeddings are fine-tuned in Phase 3 while learning trajectory embeddings. For the two baselines, Raw Trajectory and Raw Segments, we adopt the same structure of encoder–decoder used in Trembr, where the encoder and decoder are both a single layer LSTM model. For trajectory2vec and t2vec, we tune the best parameter settings, e.g., the cell size of t2vec is 100 m. The initial learning rate for all models is set to 0.0001 and we use Adam for optimization. To obtain converged results, the number of iterations for model training varies for individual models and different datasets.

5.2.2 Parameters Settings in Trembr. Here we examine the impact of parameter settings in Trembr on the result of trajectory representation learning and the performance of its applications. We only show the results of Trembr$_{f+b}$, because Trembr$_f$ and Trembr$_b$ have a similar trend with Trembr$_{f+b}$. To test the parameter sensitivity of Trembr, we vary the values of important parameters to observe how the mean rank changes, as shown in Figure 11.

Dimensionality of road segment embeddings. First, Figure 11(a) shows that setting the dimensionality of road segment embeddings at 128 for both Porto and Tokyo datasets achieves the best performance. Generally speaking, a small dimensionality does not sufficiently capture the information embedded in the relationships between road segments, but a large dimensionality may lead to noises and cause overfitting.

Window size in Road2Vec. As shown in Figure 11(b), the performance does not change much when the window size of `Moving time` in Road2Vec is set at between 45 s to 75 s. Thus, setting the window size of `Moving time` to 60 s is a good choice.

Dimensionality of trajectory embeddings. Figure 11(c) shows that setting the dimensionality of trajectory embeddings at 256 is reasonable. As mentioned previously, setting a small dimensionality does not sufficiently capture the information embedded in trajectories. Again, a large dimensionality may lead to noises and cause overfitting. For both Porto and Tokyo datasets, increasing the dimensionality from 64 to 256 significantly improves the performance (57.7% and 62.6%, respectively). While increasing dimensionality up to 512 continues to improve the performance, their improvements are relatively small (4.5% and 19.0%, respectively).

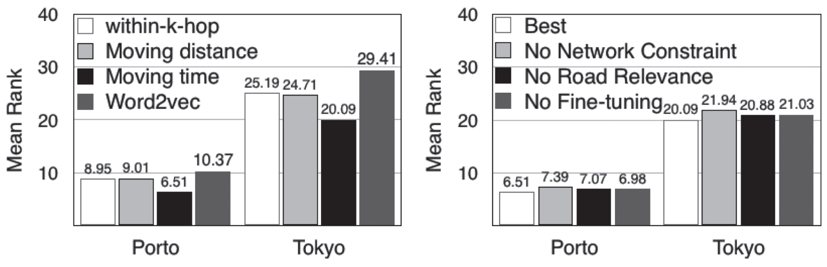
Number of iterations in Traj2Vec. In Traj2Vec, while the default batch size of each iteration is 64 (limited by the memory of GPU), Figure 11(d) suggests that when the number of iterations is increased (resulting in more data for training), the performance continues to improve and converge when it is set to 6,000 to 10,000.

Based on these results, in Trembr, the dimensionality of road segment embeddings and trajectory embeddings is set to 128 and 256, respectively. The window size of `Moving time` in Road2Vec is set to 60 s. The number of iterations for training in Traj2Vec is set to 6,000 for Porto dataset and 8,000 to Tokyo dataset while the batch size is set to 64. There are several additional parameters in Trembr that are empirically decided in the process of tuning the above parameters. Due to the lack of space, we skip the empirical details in tuning those parameters. Note that the α for weighting the loss in Road2Vec is set to 0.9, indicating that the co-occurrence relationship between road segments is more important than the same road type relationship.

5.2.3 Study of Unique Issues in Trembr. As discussed, several unique issues arise in the design of Trembr. In this section, we examine the following issues: (i) the window size for data preparation in Road2Vec, (ii) whether it is helpful to embed the network topology constraint for learning in Traj2Vec, and (iii) whether it is helpful to embed the road segment relevance for training in Traj2Vec. We perform experiments to compare alternative choices in these issues and justify our decisions.

Regarding the issue of how to define the window size for data preparation in Road2Vec, we compare three different approaches: (1) `within-`k`-hop` in the road network, where k is set to 5, (2) `Moving distance`, which is set to 500 m, and (3) `Moving time`, which is set to 60 s, while the number of negative samples per positive sample in Road2Vec is 5. We also show one additional approach (4) `Word2Vec`, which simply applies the Word2vec model [10, 25] (i.e., it uses within-k-hop to define the window size and only considers the frequent co-occurrence relationship among road segments for learning), where k is also set to 5. These values are empirically decided in the process of parameter tuning shown in the previous section. Figure 12(a) shows that using `Moving time` to define the window size is better than using the other two in both datasets (improved about 23.0% to 38.5%), because it captures the moving behaviors of mobile users, i.e., two road segments passed by within a short time are relevant. In contrast, both `within-`k`-hop` and `Moving distance` fail to capture the moving behaviors of mobile users effectively. Additionally, these three approaches all outperform `Word2vec`, which suggests that considering the same road type relationship among road segments helps further trajectory representation learning.

Regarding the issue of whether it is helpful to embed the network topology constraint and embed the road segment relevance in Traj2Vec for trajectory representation learning, we compare three settings: (1) the `Best` setting embeds both the network topology constraint and the road seg-



(a) Window size in Road2Vec (b) Issues in Traj2Vec

Fig. 12. Comparison of approaches to issues.

Table 2. Performance of Trajectory Similarity Measure

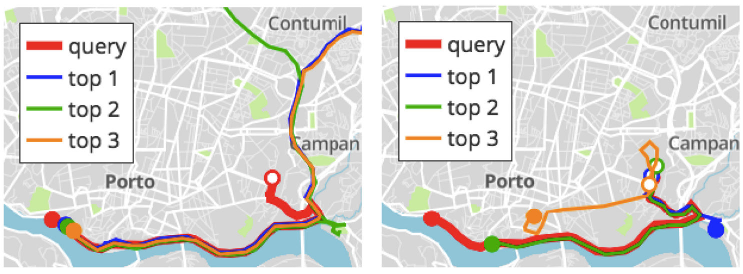
Dataset	Porto				Tokyo			
r	0.0	0.2	0.4	0.6	0.0	0.2	0.4	0.6
Raw Trajectory	256.87	1266.39	1907.45	2636.31	382.13	1501.75	2119.57	2666.31
Road Segment	163.31	340.47	1392.99	2338.04	328.1	775.81	1191.41	1585.88
trajectory2vec	367.54	1588.32	2350.23	x	447.01	1192.66	1882.23	x
t2vec	301.83	533.97	1023.42	1924.82	391.49	813.92	1074.74	1395.18
Trembr$_f$	15.07	246.21	679.73	1651.71	24.09	294.11	657.99	1176.43
Trembr$_b$	11.12	194.58	658.05	1514.53	23.49	290.34	647.65	1159.13
Trembr$_{f+b}$	**6.51**	**192.77**	**523.96**	**1201.5**	**20.09**	**281.48**	**616.58**	**1009.77**

ment relevance in the loss function of Traj2Vec, and fine-tunes the road segment embeddings; (2) the `No Network Constraint` does not consider the network topology constraint in Traj2Vec; (3) the `No Road Relevance` does not consider the road segment relevance in Traj2Vec; and (4) the `No Fine-tuning` does not fine-tune the road segment embeddings while learning trajectory embeddings. Figure 12(b) shows that `Best` outperforms `No Network Constraint` about 9.2% to 13.5%, outperforms `No Road Relevance` about 3.9% to 8.6% and outperforms `No Fine-tuning` about 4.7% to 7.2%. This suggests that considering the network topology constraint and the road segment relevance in Traj2Vec are both necessary, and the fine-tuning also helps to learn better trajectory embeddings.

5.2.4 Evaluation of Models. The performance of trajectory similarity measure using trajectory embeddings obtained by the evaluated models is summarized in Table 2. Here, we also study the impact of low and non-uniform sampling rate by down-sampling GPS sample points of trajectories. We control the dropping rate r from 0.0 to 0.6 to randomly drop GPS sample points of trajectories in the training dataset D_a and test datasets Q_a and Q_b.[4] One existing work, trajectory2vec, fails when r is 0.6, because many trajectories have too few sample points to get aggregated moving features. As shown, the Trembr variants outperform all the baselines and state-of-the-art models. We make the following observations from the comparison.

Transforming trajectories into ST-Seqs is useful. Comparing with Raw Trajectory, trajectory2vec and t2vec (which use the raw GPS sample points, aggregated moving features and cells in the map, respectively), the models using ST-Seqs achieve better performance under various r, because transforming trajectories into ST-Seqs is useful for handling noisy sample points and the

[4]With $r = 0.0$ the original D_a, Q_a, and Q_b are used (without dropping sample points).



(a) Trembr$_f$ (b) Trembr$_b$

Fig. 13. Comparison of similar trajectories.

issue of low and non-uniform sample rate. Moreover, ST-Seqs provide both spatial and temporal properties in trajectories for further trajectory representation learning, which leads Trembr outperform trjactory2vec and t2vec while trajectory2vec misses the spatial properties and t2vec misses the temporal properties. Note that in our evaluation the t2vec shows a much worse performance than its reported result [18]. This may be due to the use of much larger datasets (e.g., 10× for Porto data) in our evaluation, which is consistent with the worsen performance of t2vec observed in Reference [18], when the size of dataset is increasing.

Replacing road segment IDs by road segment embeddings is useful. Comparing with Road Segment and Trembr variants (which all use the transformed ST-Seqs with road segment embeddings), the performance improvement of Trembr variants over Road Segment is clear and impressive. It shows that replacing road segment IDs by their embeddings is critical to the success of Trembr, because the road segment embeddings capture the additional information of co-occurrence in trajectories and the same road type in road network.

Combining forward and backward information of trajectories is helpful. Among Trembr variants, Trembr$_f$ and Trembr$_b$ have close performance, while Trembr$_{f+b}$ outperforms them, which indicates that using directional information, forward and backward, of trajectories is helpful. To further investigate the reasons, we perform a search for similar trajectories to a query trajectory T_a. Figure 13(a), Figure 13(b), and Figure 1 show the top three most similar trajectories to T_a (marked as "query")-based on trajectory embeddings obtained by Trembr$_f$, Trembr$_b$, and Trembr$_{f+b}$, respectively. Here, we do not show the ground truth, i.e., the corresponding trajectory T_b of T_a, for better observation, but note that T_b is ranked at the top in this case. Figure 13(a) shows that, for Trembr$_f$, the top 3 most similar trajectories have the sources (marked by solid circles) close to each other and tend to have the first halves of the trajectories similar with the query. In other words, the learned trajectory embeddings of Trembr$_f$ mainly capture the first halves of the trajectories but miss the information of the remaining trajectories. However, Figure 13(b) shows that, for Trembr$_b$, the top three similar trajectories have the destinations (marked by hollow circles) close to each other and tend to have the second halves of the trajectories similar with the query, i.e., the learned trajectory embeddings of Trembr$_b$ miss the information of the first halves of the trajectories. Finally, Figure 1 shows that the top three similar trajectories of Trembr$_{f+b}$ have both sources and destinations close to that of the query trajectory and have trajectories similar as a whole, because Trembr$_{f+b}$ is able to utilize both forward and backward information of trajectories.

The issue of low sample rate is challenging. Although the mean rank of all approaches increases remarkably when the dropping rate r is increased, the Trembr variants consistently outperform the other methods significantly. However, one potential issue is that the Trembr variants depend on road network matching to transform raw trajectories to ST-Seqs. When the dropping

Table 3. Performance of Travel Time Prediction

Dataset	Porto				Tokyo			
r	0.0	0.2	0.4	0.6	0.0	0.2	0.4	0.6
Raw Trajectory	278.03	293.85	318.05	319.99	141.79	144.19	148.54	169.49
Road Segment	211.04	211.69	226.27	232.32	129.18	131.96	134.59	134.6
trajectory2vec	223.56	231.77	259.9	x	156.56	157.89	157.81	x
t2vec	300.15	300.73	301.56	305.14	198.67	200.41	202.57	202.79
Trembr$_f$	175.5	179.31	191.61	199.64	101.32	102.2	103.42	111.31
Trembr$_b$	177.4	183.96	192.78	203.62	102.02	102.67	103.18	107.01
Trembr$_{f+b}$	**171.97**	**175.12**	**179.01**	**189.43**	**91.78**	**92.56**	**92.76**	**96.46**

rate is large, each pair of test trajectories, T_a and T_b, have interleaving but sparse sample points, leading to potentially different ST-Seqs and thus resulting in different trajectory embeddings.

5.3 Travel Time Prediction

In this section, we demonstrate that the trajectory embeddings learned by Trembr can be used effectively for another application, travel time prediction. In the following, we first introduce the experimental setup including the experimental flow of travel time prediction and metrics for evaluation. Then, we show the experimental results of Trembr, in comparison with other models.

5.3.1 Experimental Setup. Travel time prediction is a regression task to predict the travel time of a trajectory. Given a trajectory dataset D, we randomly select 10,000 trajectories to form a test dataset Q. We use D to train the representation models and infer the embeddings for each trajectory in Q. Then, we apply a linear regression model, which uses the trajectory embeddings of Q learned by the various trajectory representation learning models under comparison as input feature vectors, with fivefold cross validation. We use *mean absolute error (MAE)* between the predicted result and the ground truth (in seconds) as the metric for travel time prediction.

Regarding the parameter settings, we use the same parameter values used in the experiments for trajectory similarity measure. In addition, for fair comparison, in Trembr variants, we do not use any time information that may reflect the prediction target. Thus, we remove travel time while encoding and decoding in Traj2Vec. Moreover, we use within-5-hop as the window size in Road2Vec rather than use Moving time.

5.3.2 Evaluation of Models. The performance of travel time prediction by all evaluated models is summarized in Table 3. As in the experiments of trajectory similarity measure, we study the impact of low and uniform sampling rate by randomly dropping some GPS sample points of trajectories. We vary the dropping rate r from 0.0 to 0.6 and randomly drop GPS sample points of trajectories in the training dataset D and test dataset Q based on the dropping rate. Like before, trajectory2vec fails when r is 0.6 as some trajectories have too few sample points to get aggregated moving features. As shown, Trembr variants outperform all the baselines and state-of-the-art models. We have the following observations from the comparison.

ST-Seqs transformation and road segment embeddings are useful. Among all approaches, Trembr variants achieve significantly better performance than Raw Trajectory, trajectory2vec and t2vec under various settings of r. It shows that transforming trajectories into ST-Seqs is useful, achieving better performance than using raw GPS sample points, aggregated moving features, or grid cells in the map, because transforming trajectories into ST-Seqs based on road network matching effectively handles noisy sample points and the issue of low and non-uniform sample

Table 4. Performance of Destination Prediction

Dataset	Porto				Tokyo			
r	0.0	0.2	0.4	0.6	0.0	0.2	0.4	0.6
Raw Trajectory	2.45	2.56	2.67	3.09	3.57	3.75	3.95	4.51
Road Segment	1.63	1.67	1.73	1.81	3.05	3.23	3.31	3.49
trajectory2vec	2.25	2.37	3.16	x	4.41	4.89	5.14	x
t2vec	1.87	1.88	2.03	2.33	3.28	3.45	3.71	3.89
Trembr$_f$	1.05	1.06	1.07	1.12	2.51	2.63	2.69	2.89
Trembr$_b$	0.99	1.01	1.03	1.1	2.19	2.3	2.45	2.67
Trembr$_{f+b}$	**0.89**	**0.94**	**0.98**	**1.04**	**1.98**	**2.18**	**2.33**	**2.51**

rate. However, Trembr variants outperform Road Segment. It shows that replacing road segment IDs by their embeddings is useful, because the road segment embeddings capture relationships of co-occurrence in trajectories and sample road type in the road network, which better present the moving behaviors of mobile users and properties of the road network.

Trembr variants have similar performance. Among Trembr variants, Trembr$_f$ and Trembr$_b$ achieve close performance, while Trembr$_{f+b}$ is slightly better than them (above 2.1% to 5.4% by varying r). One possible reason is that, although in experiments of trajectory similarity measure, Trembr$_{f+b}$ is able to capture the whole trajectories by concatenating the embeddings learned by Trembr$_f$ and Trembr$_b$, there could still be better ways to encode the whole trajectories into embeddings instead of concatenation. However, as shown in Figure 1, although the top similar results are quite matched with the query in terms of the moving paths on the map, their sources and destinations still have some gaps, which lead to different travel time. In other words, trajectories having similar embeddings may not perfectly reflect that they have similar travel time, although Trembr variants already outperform other approaches.

5.4 Destination Prediction

In this section, we demonstrate the learned trajectory embeddings can be used effectively for the third application, destination prediction. We first introduce the experimental setup and metrics for evaluation. Then, we show the experimental results of Trembr, in comparison with other models.

5.4.1 Experimental Setup. Destination prediction, which is also a regression task, aims to predict the latitude and longitude of the destination in each trajectory. In specific, given a trajectory dataset D, for each trajectory $T \in D$, we create a sub-trajectory by taking GPS sample points from the first four-fifths of the trajectory, denoted as T', to form a dataset D'. Then, we randomly select 10,000 trajectories from D' to form a test dataset Q'. We use D' to train the representation models and infer the embeddings for each trajectory in Q'. Then, We train two linear regression models, which use the trajectory embeddings in Q' learned by the trajectory representation models under comparison as input feature vectors, to predict the latitude and longitude of the destination in each trajectory in Q', with fivefold cross validation. We calculate *mean absolute error (MAE)* of the geographic distance between the predicted result and the destination in kilometers as the metric for destination prediction. Regarding default settings, we use the same parameter values used in trajectory similarity measure experiments.

5.4.2 Evaluation of Models. The performance of travel destination by all evaluated models is summarized in Table 4. We also study the impact of low and uniform sampling rate by randomly dropping some GPS sample points of trajectories in the training dataset D' and test dataset Q'

based on the dropping rate r varied from 0.0 to 0.6. Again, trajectory2vec fails when r is 0.6, because some trajectories have too few sample points to get aggregated moving features. As shown, Trembr variants outperform all the baselines and state-of-the-art models. We have the following observations from the comparison.

ST-Seqs transformation and road segment embeddings are useful. Again, Trembr variants achieve significantly better performance than other approaches under various settings of r. Moreover, it demonstrates that transforming trajectories into ST-Seqs based on road network matching and replacing road segment IDs by their embeddings learned in Road2Vec are both useful.

The second-half information of trajectories is more discriminative for destination prediction than the first-half one. Compared with Trembr$_f$, Trembr$_b$ has better performance (about 5% while r is 0.0), which shows that the second-half information of trajectories is more effective for destination prediction although Trembr$_f$ still achieve impressively better performance than other approaches. Finally, combining both-half the first and second-half information of trajectories, Trembr$_{f+b}$ further improves the performance.

6 CONCLUSIONS

This study focuses on representation learning of trajectories. Prior works fail to capture both the spatial and temporal properties inherent in trajectories and do not consider the underlying road networks even though most of the trajectory datasets are generated by capturing movement of mobile users on roads. To fill in this gap, we propose *Trembr*, a novel trajectory representation learning framework. In this framework, we design a novel RNN-based encoder–decoder model, *Traj2Vec*, to encode the spatial and temporal properties in trajectories into trajectory embeddings while constraining the learning process by exploiting underlying road networks. Moreover, we design a new neural network–based model, *Road2Vec*, to learn road segment embeddings in road networks that capture various relationships amongst road segments to facilitate the trajectory representation learning. Empirically, we demonstrate that the proposed Trembr framework is able to automatically learn embeddings for trajectory data to support a variety of trajectory applications, including trajectory similarity measure, travel time prediction and destination prediction in multiple real-world trajectory datasets. Experimental results show that Trembr soundly outperforms all the compared models under various settings.

REFERENCES

[1] Helmut Alt, Alon Efrat, Günter Rote, and Carola Wenk. 2003. Matching planar maps. *J. Algor.* 49, 2 (2003), 262–283.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 8 (2013), 1798–1828.

[3] Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. 2005. On map-matching vehicle tracking data. In *Proceedings of the International Conference on Very Large Data Bases*. VLDB Endowment, 853–864.

[4] Taxi Service Trajectory (TST) Prediction Challenge. 2015. Taxi Trajectory Prediction. Retrieved from http://www.geolink.pt/ecmlpkdd2015-challenge/.

[5] Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Doha, Qatar, 1724–1734.

[6] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 1724–1734.

[7] Jan K. Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.). Curran Associates, Inc., 577–585.

[8] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*. ACM, 135–144.

[9] Tao-yang Fu, Wang-Chien Lee, and Zhen Lei. 2017. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *Proceedings of the ACM on Conference on Information and Knowledge Management.* ACM, 1797–1806.

[10] Qiang Gao, Fan Zhou, Kunpeng Zhang, Goce Trajcevski, Xucheng Luo, and Fengli Zhang. 2017. Identifying human mobility via trajectory embeddings. In *Proceedings of the IJCAI*, Vol. 17. 1689–1695.

[11] Chong Yang Goh, Justin Dauwels, Nikola Mitrovic, Muhammad Tayyab Asif, Ali Oran, and Patrick Jaillet. 2012. Online map-matching based on hidden markov model for real-time traffic sensing applications. *IEEE Trans. Intell. Transport. Syst.* (2012), 776–781.

[12] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*. ACM, 855–864.

[13] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neur. Comput.* 9, 8 (1997), 1735–1780.

[14] Xiao Huang, Jundong Li, and Xia Hu. 2017. Label informed attributed network embedding. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*. ACM, 731–739.

[15] Ryan Kiros, Yukun Zhu, Ruslan R. Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of the Annual Conference on Neural Information Processing Systems*. 3294–3302.

[16] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the International Conference on Machine Learning*. 1188–1196.

[17] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. 2017. Attributed network embedding for learning in a dynamic environment. In *Proceedings of the ACM on Conference on Information and Knowledge Management.* ACM, 387–396.

[18] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S. Jensen, and Wei Wei. 2018. Deep representation learning for trajectory similarity computation. In *Proceedings of the International Conference on Data Engineering*. IEEE, 617–628.

[19] Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. In *Proceedings of the Annual Conference of the International Speech Communication Association*. 685–689.

[20] Siyuan Liu, Yunhuai Liu, Lionel M Ni, Jianping Fan, and Minglu Li. 2010. Towards mobility-based clustering. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*. ACM, 919–928.

[21] Siyuan Liu, Shuhui Wang, Kasthuri Jayarajah, Archan Misra, and Ramayya Krishnan. 2013. TODMIS: Mining communities from trajectories. In *Proceedings of the ACM International Conference on Information and Knowledge Management.* ACM, 2109–2118.

[22] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. 2009. Map-matching for low-sampling-rate GPS trajectories. In *Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems.* ACM, 352–361.

[23] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. 2015. Traffic flow prediction with big data: A deep learning approach. *IEEE Trans. Intell. Transport. Syst.* 16, 2 (2015), 865–873.

[24] Sebastian Mattheis. 2016. Barefoot. Retrieved from https://github.com/bmwcarit/barefoot/.

[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Annual Conference on Neural Information Processing Systems.* 3111–3119.

[26] Reham Mohamed, Heba Aly, and Moustafa Youssef. 2017. Accurate real-time map matching for challenging environments. *IEEE Trans. Intell. Transport. Syst.* 18, 4 (2017), 847–857.

[27] Paul Newson and John Krumm. 2009. Hidden Markov map matching through noise and sparseness. In *Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems.* ACM, 336–343.

[28] Anastasios Noulas, Salvatore Scellato, Neal Lathia, and Cecilia Mascolo. [n.d.]. Mining user mobility features for next place prediction in location-based services. In *Proceedings of the IEEE International Conference on Data Mining*. IEEE, 1038–1043.

[29] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Trans. Aud. Speech Lang. Process.* 24, 4 (2016), 694–707.

[30] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing*, Vol. 14. 1532–1543.

[31] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*. ACM, 701–710.

[32] Sayan Ranu, P. Deepak, Aditya D. Telang, Prasad Deshpande, Sriram Raghavan, et al. 2015. Indexing and matching trajectories under inconsistent sampling rates. In *Proceedings of the IEEE International Conference on Data Engineering*. IEEE, 999–1010.

[33] Hyun Oh Song, Stefanie Jegelka, Vivek Rathod, and Kevin Murphy. 2017. Deep metric learning via facility location. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE.

[34] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. 2016. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 4004–4012.

[35] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. 2015. Unsupervised learning of video representations using lstms. In *Proceedings of the International Conference on Machine Learning*. 843–852.

[36] Han Su, Kai Zheng, Haozhou Wang, Jiamin Huang, and Xiaofang Zhou. 2013. Calibrating trajectory data for similarity-based analysis. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM, 833–844.

[37] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems*. 3104–3112.

[38] Yoshihide Sekimoto, Takehiro Kashiyama, and Yanbo Pang. 2017. An open dataset for Tokyo trajectory. Retrieved from https://github.com/sekilab/OpenPFLOW.

[39] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale information network embedding. In *Proceedings of the International Conference on World Wide Web*. ACM, 1067–1077.

[40] Yilun Wang, Yu Zheng, and Yexiang Xue. 2014. Travel time estimation of a path using sparse trajectories. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*. ACM, 25–34.

[41] Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. 2016. A survey of transfer learning. *J. Big Data* 3, 1 (2016), 9.

[42] Christopher E. White, David Bernstein, and Alain L. Kornhauser. 2000. Some map matching algorithms for personal navigation assistants. *Transport. Res. Part C: Emerg. Technol.* 8, 1–6 (2000), 91–108.

[43] Hao Wu, Ziyang Chen, Weiwei Sun, Baihua Zheng, and Wei Wang. 2017. Modeling trajectories with recurrent neural networks. In *Proceedings of the International Joint Conferences on Artificial Intelligence*. 3083–3090.

[44] Di Yao, Chao Zhang, Zhihua Zhu, Jianhui Huang, and Jingping Bi. 2017. Trajectory clustering via deep representation learning. In *Proceedings of the International Joint Conference on Neural Networks*. IEEE, 3880–3887.

[45] Yu Zheng. 2015. Trajectory data mining: An overview. *ACM Trans. Intell. Syst. Technol.* 6, 3 (2015), 29.

Received February 2019; revised June 2019; accepted September 2019